Fast tensor-product solvers.

Part I: Partially deformed three-dimensional domains

Tormod Bjøntegaard* , Yvon Maday† , and Einar M. Rønquist‡

May 4, 2007

Abstract

We consider the numerical solution of partial differential equations in partially deformed three-dimensional domains in the sense that a *general* two-dimensional cross section in the xy-plane is invariant with respect to the z-direction. Earlier work has exploited such geometries by approximating the solution as a truncated Fourier series in the z-direction. In this paper we propose a new solution algorithm which also exploits the tensor-product feature between the xy-plane and the z-direction. However, the new algorithm is not limited to periodic boundary conditions, but works for general Dirichlet and Neumann type of boundary conditions. The proposed algorithm also works for problems with variable coefficients as long as these can be expressed as a separable function with respect to the variation in the xy-plane and the variation in the z-direction. For most problems where the new method is applicable, the computational cost is better or at least as good as the best iterative solvers. The new algorithm is easy to implement, and useful, both in a serial and parallel context. Numerical results are presented for three-dimensional Poisson and Helmholtz problems using both low order finite elements and high order spectral element discretizations.

Key words: fast solver, tensor-product, partial differential equation, parallel algorithm.

Suggested running head: Fast tensor-product solvers in space.

*Norwegian University of Science and Technology, Department of Mathematical Sciences, Trondheim, Norway

†Laboratoire Jacques-Louis Lions, Université Pierre et Marie Curie, Paris, France

‡Norwegian University of Science and Technology, Department of Mathematical Sciences, mailing address: N-7491 Trondheim, Norway, fax: +47 73 59 35 24, phone: +47 73 59 35 47, email: ronquist@math.ntnu.no

1 Introduction

Fast tensor-product solvers were proposed by Lynch, Rice and Thomas in 1964; see [10]. This is a very specialized class of solution methods with limited applicability, but they are very attractive to use whenever they are applicable. They have typically been used in the context of solving the system of algebraic equations resulting from discretized partial differential equations with constant coefficients in very simple computational domains (rectangular, hexahedral, or cylindrical domains). For example, the solution of the Poisson equation or the biharmonic equation in a cube can be done extremely efficiently using this approach; see [13, 14, 4, 1]. This class of solution methods have also been exploited in the context of constructing efficient preconditioners for iterative methods; see for example [5, 16].

Recently, the tensor-product approach has also been extended to solve elliptic problems associated with layered media; see [9]. However, this two- and three-dimensional direct solver still assumes a fully separable elliptic problem in order to reduce the multi-dimensional problem to a sequence of one-dimensional problems.

In this work we exploit the tensor-product feature to solve problems in partially tensor-product domains. In particular, we consider three-dimensional generalized cylinders (meaning with a possibly nonquadratic cross-section, and even including holes); see Figure 1. The two-dimensional cross-section may be discretized in an unstructured manner, or may be discretized in a structured manner but deformed so as to prevent fast tensor-product solvers to be used directly. The main idea is to exploit the tensor-product feature between the two-dimensional cross-section (the xy-plane) and the perpendicular z-direction.

We demonstrate the approach by solving the Poisson problem and the Helmholtz problem in selected three-dimensional domains. The proposed method results in a set of two-dimensional Helmholtz-type problems, one for each cross-sectional plane, which can be solved completely independently. The algorithm is therefore highly parallel. The two-dimensional systems can be solved either iteratively or using a direct method. Each individual two-dimensional system may also be parallelized, thus allowing for a two level parallelization: a parallelization with respect to all the two-dimensional planes, and a separate and independent parallelization of each individual plane.

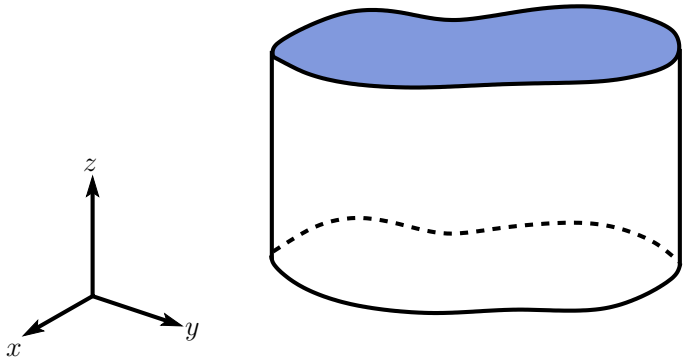

Figure 1: A two-dimensional deformed domain extended in the third direction.

2 The Poisson problem

Let Ω be a three-dimensionl domain which can be considered as a "cylinder" in the sense that a two-dimensional cross section, $\mathcal{O}$, in the xy-plane is invariant with respect to the z-direction; see Figure 1. Hence, we can write

$$\Omega = \mathcal{O} \times (0, L), \tag{2.1}$$

where L is the length of the domain in the z-direction.

We consider now the Poisson problem in such a cylinder,

$$\begin{aligned} -\nabla^2 u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\mathcal{O} \times [0, L]. \end{aligned} \tag{2.2}$$

The boundary conditions on the planes $z = 0$ and $z = L$ will be discussed below.

Earlier work has exploited domains that can be expressed as (2.1). For example, the work in [3] and [7] exploit this fact in the numerical solution of the three-dimensional Navier-Stokes equations by using a spectral element discretization in the xy-plane and a truncated Fourier expansion in the z-direction. Another example is the stability analysis of three-dimensional free surface flows; see [2]. Most earlier works trying to exploit the particular structure (2.1) have been based on models incorporating periodicity in the z-direction, i.e.,

$$u(x, y, 0) = u(x, y, L).$$

The motivation for this work is to generalize this approach, in particular, with respect to prescribed boundary conditions in the z-direction, but also with respect to more general discretizations. This is done by using diagonalization techniques between the xy-plane and the z-direction.

In the following discussion, we assume homogeneous Dirichlet boundary conditions also in the z-direction. We will return to other types of boundary conditions later.

The weak formulation of the Poisson problem (2.2) is then: Find $u \in H_0^1(\Omega)$ such that

$$a(u, v) = (f, v) \quad \forall v \in H_0^1(\Omega), \tag{2.3}$$

where the bilinear form $a(u, v)$ is given as

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega = \int_\Omega \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial v}{\partial y} + \frac{\partial u}{\partial z}\frac{\partial v}{\partial z} \right) \mathrm{d}\Omega, \tag{2.4}$$

and the right hand side is given as

$$(f, v) = \int_\Omega f v \, \mathrm{d}\Omega. \tag{2.5}$$

We will assume that the given data f is square integrable over Ω.

3 Discretization

We now discuss the numerical solution of (2.3) based on the weak formulation. The particular structure of our linear differential operator and of our domain Ω allows us to express the discrete solution space X_N as

$$X_N = \mathrm{span}\,\{\phi_m(x, y)\psi_n(z)\}, \quad m = 1, \ldots, N_2, \quad n = 1, \ldots, N_1. \tag{3.1}$$

Each basis function can thus be expressed as a product of a two-dimensional function, ϕ_m, defined over $\mathcal{O}$ and a one-dimensional function, ψ_n, defined on the interval $(0, L)$. We note that, as the resolution is increased (i.e., as the discretization parameters N_2 and N_1 go to infinity), the

two-dimensional functions $\{\phi_m(x,y)\}_{m=1}^{\infty}$ will span $H_0^1(\mathcal{O})$, while the one-dimensional functions $\{\psi_n(z)\}_{n=1}^{\infty}$ will span $H_0^1((0,L))$.

The discrete problem can be stated as: Find $u_N \in X_N$ such that

$$a(u_N, v) = (f, v) \quad \forall v \in X_N. \tag{3.2}$$

The discrete solution can be expressed in tensor product form as

$$u_N(x,y,z) = \sum_{m=1}^{N_2} \sum_{n=1}^{N_1} u_{mn}\, \phi_m(x,y)\psi_n(z), \tag{3.3}$$

where u_{mn} are the basis coefficients. Note that each basis function is a separable function while u_N is not.

Inserting (3.3) into (3.2) and choosing test functions $v(x,y,z) = \phi_i(x,y)\psi_j(z)$, we get

$$\sum_{m=1}^{N_2} \sum_{n=1}^{N_1} \left[(\int_{\mathcal{O}} \widetilde{\nabla}\phi_i \cdot \widetilde{\nabla}\phi_m \, dx\, dy)(\int_0^L \psi_j\psi_n dz) + (\int_{\mathcal{O}} \phi_i\phi_m \, dx\, dy)(\int_0^L \psi_j'\psi_n' dz) \right] u_{mn} = g_{ij}. \tag{3.4}$$

Here, $\widetilde{\nabla}$ denotes the gradient operator in two space dimensions (in the xy-plane), and

$$g_{ij} = \int_0^L \left(\int_{\mathcal{O}} f(x,y,z)\phi_i(x,y)\, dx\, dy \right) \psi_j(z)\, dz. \tag{3.5}$$

In the case of Dirichlet boundary conditions in the z-direction, $\psi_n(0) = \psi_n(L) = 0$ for all $n = 1, \ldots, N_1$. A convenient choice of functions $\psi_n(z)$ are those functions that satisfy

$$\int_0^L \psi_j'\psi_n' dz = \lambda_n \int_0^L \psi_j\psi_n dz, \qquad \begin{array}{l} \forall j = 1, \ldots, N_1, \\ \forall n = 1, \ldots, N_1. \end{array} \tag{3.6}$$

where λ_n is a constant. Inserting (3.6) into (3.4), we arrive at

$$\sum_{m=1}^{N_2} \sum_{n=1}^{N_1} \left[\left(\int_{\mathcal{O}} \left(\widetilde{\nabla}\phi_i \cdot \widetilde{\nabla}\phi_m + \lambda_n \phi_i\phi_m \right) dx\, dy \right) \left(\int_0^L \psi_j\psi_n dz \right) \right] u_{mn} = g_{ij}. \tag{3.7}$$

On a continuous level, (3.6) is equivalent to choosing $\psi_n(z)$ to be the solution of the symmetric

eigenvalue problem

$$-\psi_n'' = \lambda_n \psi_n, \quad \psi_n(0) = \psi_n(L) = 0,$$

which is explicitly given as

$$\psi_n(z) = \sqrt{2/L}\,\sin(n\pi z/L), \tag{3.8}$$

$$\lambda_n = n^2\pi^2/L^2. \tag{3.9}$$

Note also that these eigenfunctions $\psi_n(z)$ are orthonormal,

$$\int_0^L \psi_j \psi_n dz = \delta_{jn}, \quad \forall j, n. \tag{3.10}$$

Inserting (3.6) into (3.7) and using (3.10), we arrive at the systems

$$\sum_{m=1}^{N_2} \left[\int_{\mathcal{O}} \left(\widetilde{\nabla}\phi_i \cdot \widetilde{\nabla}\phi_m + \lambda_n \phi_i \phi_m \right) dx\, dy \right] u_{mj} = g_{ij}, \qquad \begin{array}{l} \forall i = 1, \ldots, N_2, \\ \forall j = 1, \ldots, N_1. \end{array} \tag{3.11}$$

We see that this approach transforms the solution of a single three-dimensional problem into the solution of N_1 two-dimensional Helmholtz-type systems.

However, other choices for $\psi_n(z)$ are also possible. For example, in the low order finite element case, we can choose the $\psi_n(z)$ to be the eigenvectors of the one-dimensional stiffness matrix with respect to the one-dimensional mass matrix; these eigenvectors are then spanned by the usual one-dimensional "hat" basis functions. Similarly, in the high order spectral (element) case, we can choose the functions $\psi_n(z)$ to be the eigenvectors of stiffness matrix with respect to the one-dimensional mass matrix, however, now the eigenvectors would be spanned by (piecewise) high order polynomials which vanish at the end points $z = 0$ and $z = L$.

We remark that, for both the low order finite element case and the high order spectral (element) case, the discrete eigenvalues λ_n and the corresponding discrete eigenfunctions ψ_n are different from the analytical expressions (3.9) and (3.8). An expansion in the z-direction based on the analytical eigenfunctions (3.8) represents an approach similar to the Fourier expansion for periodic problems. The disadvantage with this approach is that we need to find the explicit form

for the analytical eigenfunctions (and eigenvalues) for each specific type of boundary conditions specified at $z = 0$ and $z = L$, and for each type of operator. This may be cumbersome or difficult, and the implementation will be different for each specific case. In the following, we will therefore focus on a finite element or spectral (element) discretization in the z-direction, thus allowing for a more general and flexible setting.

3.1 Algebraic system of equations

Let us now proceed with the details for arbitrary finite element discretizations, both low order and high order discretizations. We let $\phi_m(x, y)$ be any two-dimensional basis function, and let $\psi_n(z)$ be any one-dimensional basis function. The discrete space X_N and the discrete solution u_N are then given by (3.1) and (3.3), respectively. If both ϕ_m and ψ_n are nodal basis functions, the unknown basis coefficients u_{mn} will be the numerical approximation at the nodal points.

It now follows directly from (3.4) that we can express the system of algebraic equations as

$$\sum_{m=1}^{N_2}\sum_{n=1}^{N_1}(A_{im}^{2D}B_{jn}^{1D} + B_{im}^{2D}A_{jn}^{1D})\,u_{mn} = g_{ij}, \qquad \begin{matrix}\forall i = 1,\ldots,N_2,\\ \forall j = 1,\ldots,N_1.\end{matrix} \tag{3.12}$$

where

$$\begin{aligned} A_{im}^{2D} &= \int_{\mathcal{O}} \widetilde{\nabla}\phi_i \cdot \widetilde{\nabla}\phi_m \, dx\, dy, \\ B_{im}^{2D} &= \int_{\mathcal{O}} \phi_i\phi_m \, dx\, dy, \end{aligned}$$

are the elements in the two-dimensional stiffness matrix and mass matrix, respectively, and

$$\begin{aligned} A_{jn}^{1D} &= \int_0^L \psi_j'\psi_n' \, dz, \\ B_{jn}^{1D} &= \int_0^L \psi_j\psi_n \, dz, \end{aligned}$$

are the elements in the one-dimensional stiffness matrix and mass matrix associated with the z-direction. The right hand side elements g_{ij} in (3.12) are given in (3.5).

In matrix form, we can write (3.12) succinctly as

$$(B^{1D} \otimes A^{2D} + A^{1D} \otimes B^{2D})\, \underline{u} = \underline{g}, \tag{3.13}$$

where we have used standard tensor product notation. Note that $\underline{u}$ is a vector of length $N_2 N_1$, and that the unknowns are numbered in such a way that the all the nodes in a fixed xy-plane are numbered before proceeding to the next plane. A similar numbering scheme is used for the right hand side $\underline{g}$.

3.2 Diagonalization

We now consider fast tensor-product techniques to solve the system (3.13). To this end, we introduce the generalized eigenvalue problem

$$A^{1D} \underline{q}_j = \lambda_j\, B^{1D} \underline{q}_j, \qquad j = 1, \ldots, N_1,$$

or

$$A^{1D} Q = B^{1D} Q \Lambda. \tag{3.14}$$

Here, $\underline{q}_j$ is an eigenvector of the one-dimensional discrete Laplace operator A^{1D} with respect to the one-dimensional mass matrix B^{1D}, and λ_j is the corresponding (real and positive) eigenvalue. The columns of Q contain all the eigenvectors, while Λ is a diagonal matrix containing the eigenvalues along the diagonal.

Normalizing the eigenvectors with respect to B^{1D} we get

$$B^{1D} = Q^{-T} Q^{-1}, \tag{3.15}$$

$$A^{1D} = Q^{-T} \Lambda\, Q^{-1}. \tag{3.16}$$

From (3.13) and (3.15)-(3.16) it follows that

$$\begin{aligned} B^{1D} \otimes A^{2D} + A^{1D} \otimes B^{2D} &= Q^{-T} Q^{-1} \otimes A^{2D} + Q^{-T} \Lambda \, Q^{-1} \otimes B^{2D}, \\ &= (Q^{-T} \otimes I^{2D})(I^{1D} \otimes A^{2D} + \Lambda \otimes B^{2D})(Q^{-1} \otimes I^{2D}), \end{aligned}$$

where I^{2D} and I^{1D} are identity matrices of the same dimension as the number of degrees-of-freedom in each xy-plane and in the z-direction, respectively.

By introducing the variables

$$\begin{aligned} \underline{\tilde{u}} &= (Q^{-1} \otimes I^{2D}) \, \underline{u}, \\ \underline{\tilde{g}} &= (Q^{T} \otimes I^{2D}) \, \underline{g}, \end{aligned}$$

the algebraic problem can be written as

$$(I^{1D} \otimes A^{2D} + \Lambda \otimes B^{2D}) \, \underline{\tilde{u}} = \underline{\tilde{g}}.$$

In "semi-local" form (global numbering in the xy-plane, but local numbering for the combined xy-plane and the z-direction), this reduces to

$$\sum_{m=1}^{N_2} \sum_{n=1}^{N_1} \big[(A^{2D})_{im} \underbrace{(I^{1D})_{jn}}_{\delta_{jn}} + (B^{2D})_{im} \underbrace{(\Lambda)_{jn}}_{\lambda_j \delta_{jn}} \big] \tilde{u}_{mn} = \tilde{g}_{ij}, \qquad \begin{matrix} i = 1, \ldots, N_2, \\ j = 1, \ldots, N_1. \end{matrix} \tag{3.17}$$

or

$$\sum_{m=1}^{N_2} \big[(A^{2D})_{im} + \lambda_j (B^{2D})_{im} \big] \tilde{u}_{mj} = \tilde{g}_{ij}, \qquad \begin{matrix} i = 1, \ldots, N_2, \\ j = 1, \ldots, N_1. \end{matrix} \tag{3.18}$$

Here, the first index runs over the entire xy-plane, while the second index corresponds to the z-direction.

Thus, the total algorithm comprises three steps.

The first step is to transform the right hand side:

$$\tilde{g}_{ij} = \sum_{m=1}^{N_2} \sum_{n=1}^{N_1} \underbrace{(I^{2D})_{im}}_{\delta_{im}} Q^T_{jn} g_{mn} \tag{3.19}$$

$$= \sum_{n=1}^{N_1} Q^T_{jn} g_{in} = \sum_{n=1}^{N_1} g_{in} Q_{nj}, \qquad \begin{array}{l} i = 1, \ldots, N_2, \\ j = 1, \ldots, N_1. \end{array} \tag{3.20}$$

or, in matrix form,

$$\widetilde{G} \;=\; G\,Q. \tag{3.21}$$

Here, G and $\widetilde{G}$ are the given data and transformed data, respectively, in a "semi-local" data representation.

The second step is to solve the systems (3.18). Each system (for a fixed value of j) couples the degrees-of-freedom within a single xy-plane. Using a global data representation for the unknowns within each plane, the systems (3.18) can also be expressed as:

$$(A^{2D} + \lambda_j\, B^{2D})\, \underline{\tilde{u}}_j = \underline{\tilde{g}}_j, \quad j = 1, \ldots, N_1. \tag{3.22}$$

Each two-dimensional solution $\underline{\tilde{u}}_j$ forms a column in the two-dimensional solution matrix $\widetilde{U}$ following a semi-local data representation. Note that these systems represent completely decoupled two-dimensional systems which can be solved independently of each other.

The third step involves a transformation of $\widetilde{U}$ to the final solution U. Again, using a "semi-local" data representation, we get

$$u_{ij} = \sum_{n=1}^{N_1} \tilde{u}_{in} Q_{jn} = \sum_{n=1}^{N_1} \tilde{u}_{in} Q^T_{nj} \tag{3.23}$$

or, in matrix form,

$$U \;=\; \widetilde{U}\, Q^T. \tag{3.24}$$

4 Numerical results

We now present numerical results using the proposed solution algorithm to solve the resulting systems of algebraic equations. All the numerical tests we report have been implemented in MATLAB®.

4.1 Finite element discretization

We first consider the solution of the Poisson problem in a wedge-shaped domain as depicted in Figure 2. The discretization is based on linear finite elements. Each two-dimensional cross section corresponds to a sector $\pi/4$ of the unit circle, which is discretized into triangles with mesh size h. The grid spacing in the z-direction is h. Hence, our three-dimensional discretization corresponds to using prismatic linear finite elements of mesh size h.

We derive the right-hand side, f, by using the exact solution

$$u(x,y,z) = \sin\left(2\pi\left(x^2+y^2\right)\right)\sin\left(x\left(y-\tan\left(\frac{\pi}{4}\right)x\right)\pi\right)\sin\left(\frac{\pi z}{L}\right).$$

The domain length in the z-direction is $L = 1$, and homogeneous Dirichlet boundary conditions are imposed along the entire domain boundary $\partial\Omega$. Figure 3 shows the convergence results. As expected, the error between the exact solution and the finite element solution decreases as $\mathcal{O}(h^2)$ as the mesh size h decreases; the error is here measured in the discrete L^2-norm. Note that the finest grid used in the convergence study uses 90 elements in the radial direction, in the angular direction, and in the z-direction. Hence, the resulting algebraic system of equations corresponds to about 340,000 degrees-of-freedom. All the two-dimensional systems are solved directly using full LU-factorization. However, even with a simple solver that does not exploit the sparsity associated with the triangulation of each plane, the solution is readily obtained using MATLAB®.

4.2 Spectral discretization

We now demonstrate the proposed method by solving the Poisson problem in a domain Ω as depicted in Figure 4. The domain is a deformed rectangle in the xy-plane which is extended in

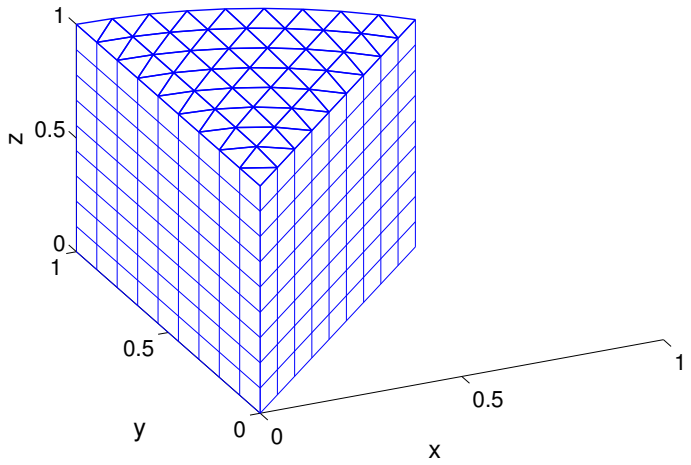

Figure 2: The finite element grid for a wedge-shaped domain.

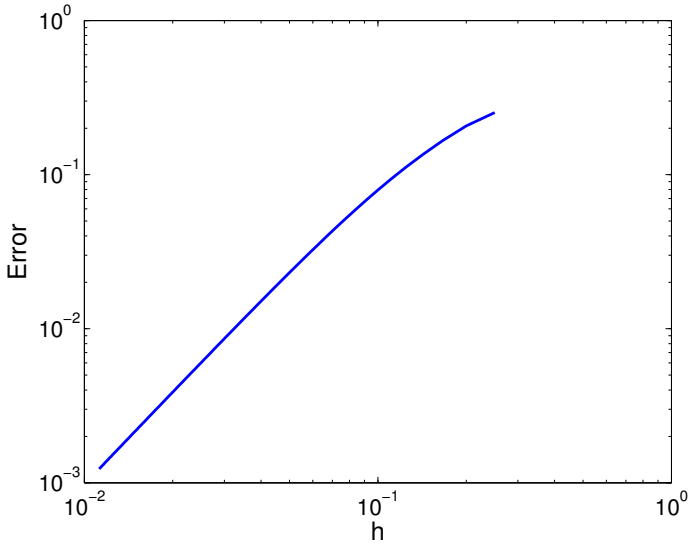

Figure 3: Relative error between the exact solution u and the numerical solution u_h of the three-dimensional Poisson problem using linear finite elements with mesh size h; the error is measured in the discrete L^2 norm. The computational grid is depicted in Figure 2.

the z-direction. The Poisson problem is discretized using a pure spectral method based on high order polynomials. Figure 4 indicates both the deformed Gauss-Lobatto Legendre grid in the xy-plane as well as a Gauss-Lobatto Legendre distribution of the nodes in the z-direction.

For the numerical experiments, we compute the right hand side, f, from the exact solution

$$u(x,y,z) = \sin\left(\frac{\pi(x - a_4\sin(\pi y))}{1 + a_2\sin(2\pi y) - a_4\sin(\pi y)}\right)\sin\left(\frac{3\pi(y + a_1\sin(2\pi x))}{1 + a_3\sin(\pi x) + a_1\sin(2\pi x)}\right)\sin\left(\frac{2\pi z}{L}\right), \tag{4.1}$$

with $a_1 = 0.08$, $a_2 = 0.10$, $a_3 = 0.12$ and $a_4 = 0.15$.

The set of algebraic equations are solved using (3.21), (3.22), and (3.24). In contrast to the finite element case, the two-dimensional problems in (3.22) are now solved iteratively using the conjugate gradient method. For the convergence result of the spectral method, we have measured the discretization error in the discrete L^2-norm. Figure 5 shows the relative error as a function of the polynomial degree, N, in each spatial direction. We see how the exponential convergence is influenced by the stopping criterion, ε, used in the conjugate gradient method for each two-dimensional problem in (3.22).

4.3 Spectral element discretization

Again, we consider the Poisson-problem with homogeneous Dirichlet boundary conditions, but the three-dimensional domain Ω is now a cylinder; see Figure 6. The two-dimensional cross section is a circle with radius equal to two. Each cross section is divided into five elements, while two layers of elements is used in the z-direction; see Figure 6.

The exact solution of the Poisson problem is

$$u(x,y,z) = \sin\left(\frac{7\pi}{4}\sqrt{(x-2)^2 + (y-2)^2} + \frac{\pi}{2}\right)\sin\left(\frac{2\pi z}{L}\right). \tag{4.2}$$

In Figure 7, we show the discretization error in the discrete L^2-error as a function of the polynomial degree, N, used to approximate the solution in each in each element. Note that the finest grid used in this convergence study corresponds to a polynomial degree $N = 29$; with ten elements, this corresponds to about 300,000 degrees-of-freedom. Similar to earlier experiments, the numerical results are obtained using MATLAB®.

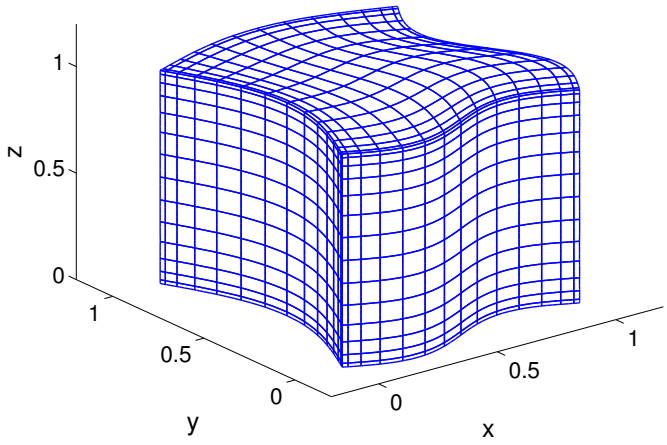

Figure 4: The domain Ω used in the numerical experiment in Section 4.2.

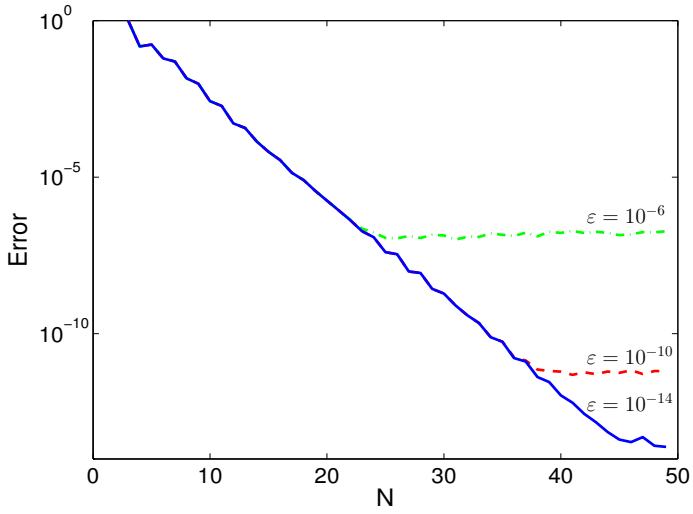

Figure 5: Relative error between the exact solution u and the numerical solution u_N of the three-dimensional Poisson problem as a function of the polynomial degree, N, in each spatial dimension. The error is measured in the discrete L^2-norm. The convergence results are shown for different choices of the stopping criterion, ε, used in the conjugate gradient iteration when solving the two-dimensional systems (3.22): $\varepsilon = 10^{-14}$, $\varepsilon = 10^{-10}$ and $\varepsilon = 10^{-6}$.

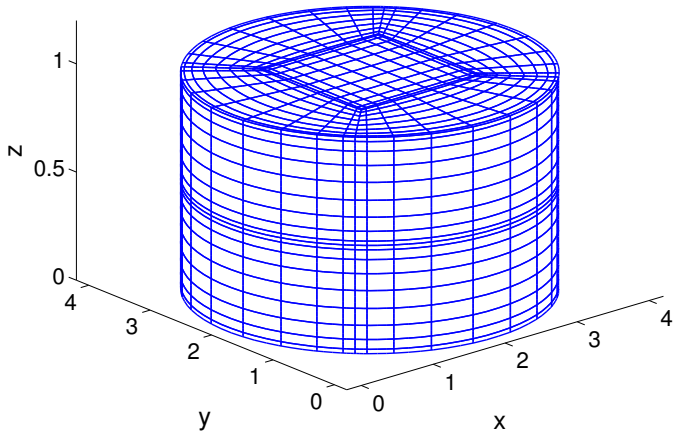

Figure 6: The computational domain, Ω, and the spectral element discretization. Two layers of elements are used in the z-direction. Each two-dimensional cross-section is divided into five spectral elements.

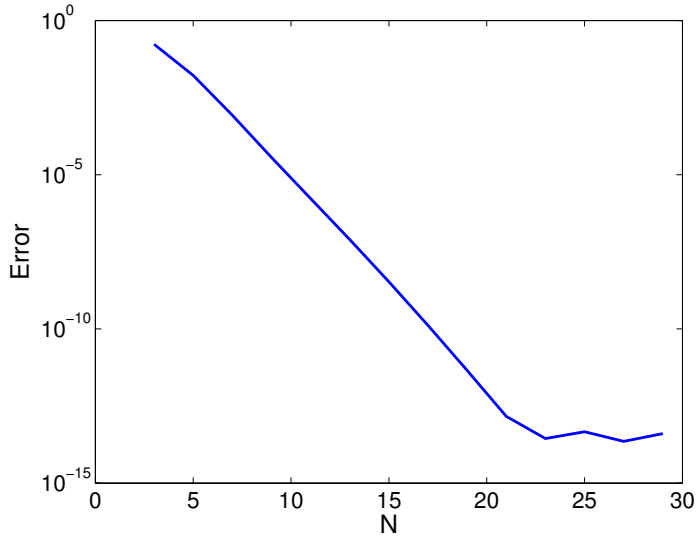

Figure 7: Relative error between the exact solution u and the numerical solution u_N of the three-dimensional Poisson problem as a function of the polynomial degree, N, in each spatial direction in each element. The error is measured in the discrete L^2-norm. The exact solution is given by (4.2). Each two-dimensional system in (3.22) is solved using conjugate gradient iteration with a tolerance $\varepsilon = 10^{-12}$.

4.4 Extension to solving the Helmholtz problem

We now extend the proposed method to solving the Helmholtz-problem

$$-\nabla^2 u + \alpha u = f \qquad \text{in } \Omega, \tag{4.3}$$

$$u = 0 \qquad \text{on } \partial\Omega, \tag{4.4}$$

where α is a positive constant. The only difference from the solution procedure for the Poisson problem is that the two-dimensional problems in (3.22) now read

$$\left(A^{2D} + (\lambda_j + \alpha)\, B^{2D}\right) \underline{\tilde{u}}_j = \underline{\tilde{g}}_j, \quad j = 1, \ldots, N_1. \tag{4.5}$$

The domain Ω and the discretization is the same as described in Section 4.2; see Figure 4. Again, we use the exact solution (4.1) to derive the right hand side, f. The parameter α is set equal to one. Figure 8 shows the convergence plot. As expected, we obtain exponential convergence similar to the Poisson problem.

4.5 Further extensions

We conclude with an example showing the extension of the proposed approach to solving problems with other types of boundary conditions and nonconstant material properties. The particular example we consider can be formulated mathematically as

$$-\nabla \cdot \kappa \nabla u = f \qquad \text{in } \Omega, \tag{4.6}$$

$$u = 0 \qquad \text{at } z = 0, \tag{4.7}$$

$$\kappa \frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\mathcal{O} \times [0, L], \tag{4.8}$$

$$\kappa \frac{\partial u}{\partial n} = q \qquad \text{at } z = L. \tag{4.9}$$

This problem describes steady heat transfer in a domain Ω with the temperature $u = 0$ prescribed on the plane $z = 0$, a heat flux, q, prescribed on the plane $z = L$, and no heat transfer through the cylinder wall. A volumetric heat source, f, is assumed given. We choose the domain Ω

and the discretization to be the same as depicted in Figure 6. The parameter κ is the thermal conductivity. As long as κ can be expressed as

$$\kappa(x, y, z) = \kappa_2(x, y) \cdot \kappa_1(z),$$

the proposed solution algorithm still applies.

We solve this test problem with $f = 0$, $q = 1$, $\kappa = 1$ in the lower half of the cylinder, and $\kappa = 2$ in the upper half of the cylinder. This problem has a simple analytical solution which only depends on z; the exact solution is piecewise linear with a jump in the first derivative at $z = 1/2$,

$$\begin{aligned} u(x, y, z) &= z \quad \text{for } 0 \leq z \leq \frac{1}{2}, \\ u(x, y, z) &= \frac{1}{2}z + \frac{1}{4} \quad \text{for } \frac{1}{2} \leq z \leq 1. \end{aligned}$$

The numerical solution is, indeed, constant in each xy-plane, and the variation in the z-direction is shown in Figure 9. We remark that using the analytic eigenfunctions (3.8) would be inappropriate for this problem due to the nonconstant thermal conductivity in the z-direction. In this sense, the proposed diagonalization procedure offers convenience in terms of handling situations with variable coefficients and other types of boundary conditions.

5 Computational complexity

We now discuss the computational cost of the proposed procedure. This includes the cost of the diagonalization step (3.14), and the solution steps (3.21), (3.22), and (3.24).

In the following, we denote the total number of elements (finite elements or spectral elements) in the domain as K, the number of elements in a two-dimensional cross-section as K_{xy}, while the number of layers of elements in the z-direction is denoted as K_z, i.e., $K = K_{xy} \cdot K_z$. In the spectral element case, we denote the polynomial order as N. Note that, in the spectral (element) case, the polynomial order need not be the same in the z-direction as in the xy-plane. With

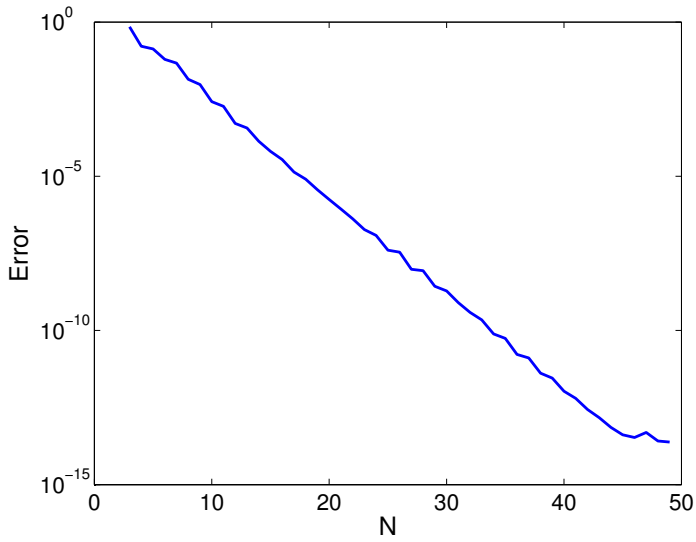

Figure 8: Relative error between the exact solution u and the numerical solution u_N of the three-dimensional Helmholtz problem (4.4) as a function of the polynomial degree, N, in each spatial direction. The error is measured in the discrete L^2-norm.

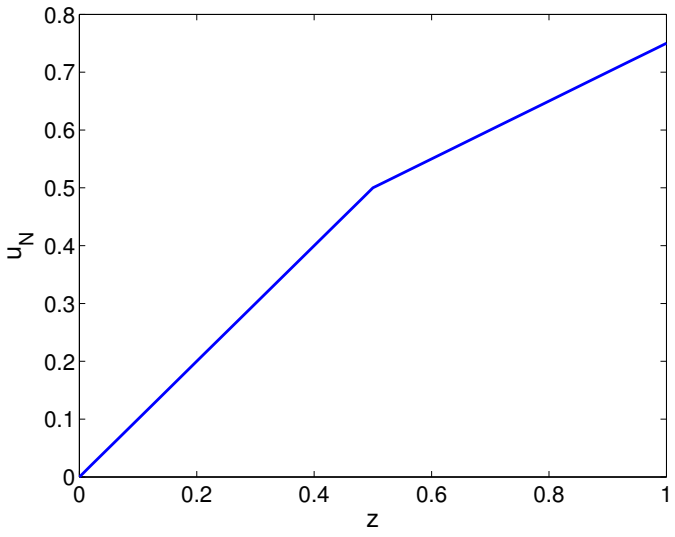

Figure 9: A plot of the variation in the numerical solution of (4.6) - (4.9) in the z-direction at a fixed point in the xy-plane. In this problem, $f = 0$ and $q = 1$. As expected, the numerical solution coincides with the exact solution.

these parameters, it follows that

$$N_2 \approx K_{xy} \qquad \text{using linear finite elements,}$$
$$N_2 \approx K_{xy} \cdot N^2 \qquad \text{using spectral elements,}$$

and

$$N_1 \approx K_z \qquad \text{using linear finite elements,}$$
$$N_1 \approx K_z \cdot N \qquad \text{using spectral elements.}$$

The exact numbers N_2 and N_1 will depend on the prescribed boundary conditions.

In order to provide a "reference" for a cost analysis of the proposed method, we first estimate the cost of performing a single operator evaluation involving the three-dimensional discrete Laplacian:

$$72\,K \qquad \text{using linear finite elements,} \tag{5.1}$$
$$12\,K\,N^4 \qquad \text{using spectral elements.} \tag{5.2}$$

In the linear finite element case, this estimate is based on an element-by-element approach, and assuming an already explicitly contructed element matrix of dimension 6×6. In the spectral element case, this estimate assumes the use of tensor product sum-factorization techniques; see [12].

We now proceed with a cost analysis of the proposed algorithm; a summary of this cost analysis is given in Table 1. The cost of solving the one-dimensional eigenvalue problem (3.14) is approximately equal to N_1^3. This estimate just states the usual fact that the cost of solving the eigenvalue problem scales like the cube of the dimension of the problem; see [6]. Let us compare this cost with the cost of performing a single operator evaluation involving the discrete Laplacian. To this end, let us first assume that $K_z \sim \sqrt{K_{xy}}$. The cost of the eigenvalue problem

	Linear finite elements	Spectral elements
N_1	K_z	$K_z N$
N_2	K_{xy}	$K_{xy} N^2$
Operator evaluation $\underline{w} = A^{3D}\underline{v}$	$72K$	$12KN^4$
Eigenvalue problem (3.14); $K_z \sim \sqrt{K_{xy}}$	K	KN^3
Transformation (3.21) and (3.24)	$4K_zK$	$4K_zKN^4$
Solution of (3.22) (M iterations)	$18MK$	$8MKN^4$
Total cost (proposed solver)	$(18M + 4K_z)K$	$(8M + 4K_z)KN^4$
Total cost (PCG, ideal 3D preconditioner)	$72MK$	$12MKN^4$

Table 1: A table showing the scalings and estimated cost for the various parts of the proposed algorithm. The total cost is also compared to an iterative solution of the full three-dimensional system, using an "ideal" preconditioner in the sense of incuring an insignificant computational cost, while giving a resolution-independent convergence rate.

is then

$$N_1^3 \sim K_z^3 \sim K \qquad \text{using linear finite elements,}$$
$$N_1^3 \sim K_z^3 N^3 \sim K N^3 \qquad \text{using spectral elements.}$$

In the linear finite element case, the cost of the eigenvalue problem will then be significantly smaller than the cost of performing a single operator evaluation (assuming we exploit all the sparsity); see (5.1). In the spectral element case, the cost of the eigenvalue problem will also be significantly less than the cost of performing a single operator evaluation since the latter scales like $\mathcal{O}(K\,N^4)$; see (5.2). If $K_z \ll \sqrt{K_{xy}}$, this conclusion will be even more true. If $K_z \gg \sqrt{K_{xy}}$, the solution of the eigenvalue problem will become more important. However, the resolution in the z-direction has to be quite extreme compared to the x- and y-direction before the eigenvalue problem will play a significant role; see below.

The transformation step (3.21) involves a matrix-matrix product. The number of floating point operations for this step is

$$2\,N_2 \cdot N_1^2 \;\approx\; 2\,K_{xy} \quad \cdot (K_z)^2 \quad = 2\,K_z \cdot K \qquad \text{using linear finite elements,} \tag{5.3}$$
$$2\,N_2 \cdot N_1^2 \;\approx\; 2\,K_{xy}\,N^2 \cdot (K_z N)^2 = 2\,K_z \cdot K\,N^4 \qquad \text{using spectral elements.} \tag{5.4}$$

Hence, for $K_z < 36$ in the linear finite elements case, and for $K_z < 6$ in the spectral element

case, the cost of (3.21) is less than the cost of a single iteration using an iterative solver for the full three-dimensional problem.

Step (3.22) involves solving approximately N_1 two-dimensional problems. The cost of these problems depends on the particular solver used. For an elliptic problem like the Poisson problem, we can use a preconditioned conjugate gradient method with a domain-decomposition-based preconditioner. In the optimal case, the cost of each two-dimensional solve will at least be a fixed number of iterations, M, times the cost of a *two-dimensional* operator evalution (or matrix-vector product); see [15]. In the linear finite element case, this lower bound can be estimated to be $18\,M\,K_{xy}$, while in the spectral element case, this lower bound is equal to $8\,M\,K_{xy}\,N^3$. The total number of floating point operations for (3.22) is then approximately equal to

$$\begin{aligned} 18\,M\,K &\qquad \text{using linear finite elements,} \\ 8\,M\,K\,N^4 &\qquad \text{using spectral elements.} \end{aligned}$$

For non-optimal preconditioners, this cost will be higher. We have here neglected the cost of the preconditioner and other aspects of the iterative solver. Note that from an implementation point of view, implementing "optimal" preconditioners in the two-dimensional case is much easier than in the three-dimensional case. Finally, note that solving two-dimensional problems has eliminated potential difficult aspect ratio problems associated with domains with a small/large length scale in the z-direction compared to a typical length scale in the xy-plane.

The above analysis only considers an iterative solution strategy for the two-dimensional problems in (3.22). This is the most practical strategy for large problems. However, it may also be possible to use a direct solution method for these two-dimensional systems even though this may not be practical for the full three-dimensional problem. We refer to the numerical test using linear finite elements in Section 4.1 as an example. If a direct method is used for the two-dimensional systems, the entire proposed solution procedure can be classified as a direct solution method.

The final transformation step (3.24) involves a matrix-matrix product at a similar cost as (3.21); see (5.3)-(5.4).

In summary, if we can neglect the cost of solving the one-dimensional eigenvalue problem,

the total cost of the proposed algorithm is then

$$\begin{aligned} &(18\,M\,+\,4\,K_z)\cdot K && \text{using linear finite elements,} \\ &(8\,M\,+\,4\,K_z)\cdot K\,N^4 && \text{using spectral elements.} \end{aligned}$$

We have here assumed iterative solution of the two-dimensional systems using optimal (ideal) preconditioners, and each system taking M iterations to converge.

Consider now the cost of using an iterative solver for the full three-dimensional problem with an optimal (ideal) preconditioner that also takes M iterations to converge. From (5.1) and (5.2) this cost is at least

$$\begin{aligned} &(72\,M)\cdot K && \text{using linear finite elements,} \\ &(12\,M)\cdot K\,N^4 && \text{using spectral elements.} \end{aligned}$$

The proposed method should therefore (conservatively) be competitive with the best (ideal) iterative solution methods if

$$\begin{aligned} &K_z < 13\,M && \text{using linear finite elements,} \\ &K_z < M && \text{using spectral elements.} \end{aligned}$$

For example, if $M = 20$, we can have up to $K_z = 260$ layers of linear finite elements, and $K_z N = 200$ nodes in the z-direction in the spectral element case (assuming $N = 10$).

Again, this is a conservative estimate. For most problems where the proposed algorithm is applicable, the computational complexity will be smaller than the best iterative solution method for the full three-dimensional problem. Additional issues in favor of the proposed approach are: elimination of the aspect ratio problem discussed earlier; the number of iterations, M, is typically larger for a full three-dimensional problem compared to solving a problem in a two-dimensional cross section; the number of iterations, M, is not a constant, but a function of the polynomial degree N in the case of using spectral elements; easier implementation (two-dimensional versus three-dimensional solvers); easier parallelization (see the next section).

6 Parallel processing

The solution step (3.22) suggests a natural way to parallelize this algorithm. One way is to simply solve one (or more) two-dimensional problem(s) on a single processor. This will imply that all the two-dimensional problems can be solved in parallel *without any communication*. The implementation of this step will be unchanged from a serial version.

If we assume that the eigenvector matrix Q is stored on all the processors, step (3.21) can be done by first storing the right hand side g_{ij} for some values of i in the xy-plane, and for all values of j along the z-direction. The transformation of the given data as given by (3.20) or (3.21) can then be done in parallel *without any communication*.

The transformed data $\widetilde{G}$ can now be "transposed" so that $\tilde{g}_{ij}$ will be available on each processor for *all* values of the index i in the xy-plane, and for one or more values of j. This "transpose" operation will require global communication.

The two-dimensional systems (3.22) can now be solved in parallel in a completely decoupled fashion, and the solution $\widetilde{U}$ will be distributed so that $\tilde{u}_{ij}$ will be available on each processor for *all* values of the index i in the xy-plane, and for one or more values of j. This distribution is again "transposed" so that $\tilde{u}_{ij}$ will be available on each processor for some values of i in the xy-plane, and for all values of j along the z-direction. The final solution U is then obtained from (3.24) by performing the matrix-matrix product in parallel *without any communication*.

Hence, the total communication cost for the algorithm is limited to the two "transpose" operations which imply an all-to-all-type of communication pattern. The parallel implementation as described above should be compared with a more standard domain decomposition approach which typically implies a parallel implementation of the preconditioned conjugate gradient method applied to the full three-dimensional system. The advantage with the proposed method is that it is has low computational complexity and it is easy to implement both in a serial and a parallel context.

It is interesting to notice that an iterative solution of the independent two-dimensional systems will enjoy both the convergence rate and ease of implementation associated with two-dimensional systems. As explained above, we can solve each two-dimensional system on a single processor. However, in the case that each two-dimensional system is also very large, we can parallelize

the solution of each two-dimensional system as well (in addition to the parallelization across two-dimensional planes). For such large problems, the proposed algorithm would offer a way to exploit computer platforms with many processors.

7 Conclusions

We have presented a new solution algorithm for problems in computational domains which can be expressed as a tensor-product between a general two-dimensional domain in the xy-plane and the z-direction. The new algorithm allows general boundary conditions to be specified in the z-direction. It can also be used to solve problems with variable coefficients as long as these can be expressed as a separable function with respect to the variation in the xy-plane and the variation in the z-direction.

For most problems where the proposed method is applicable, the computational complexity is better or at least as good as the best available iterative solvers. An attractive feature with the proposed method is that it eliminates the aspect ratio problem associated with domains which have a small length scale in the z-direction compared to a typical length scale in the xy-plane. The method also allows for an easy implementation, both in a serial and a parallel context.

We have demonstrated the new algorithm by solving selected Poisson- and Helmholtz-type problems. However, we remark that the method is equally applicable for three-dimnensional geometries with other two-dimensional topologies, e.g., a planar region with a certain number of holes, and for solving other partial differential equations.

8 Future work

Future work will include the application of the method presented here to simulate three-dimensional Bénard-Marangoni convection [8]. Previous numerical results for this problem have assumed a fixed and undeformed free surface [11]. However, it is known that the (unknown) free surface will be slightly deformed for this type of problems. In the context of solving the governing equations (the incompressible Navier-Stokes equations and the energy equation) numerically using a splitting approach, the computational problem is reduced to solving a Helmholtz-type equation

for the velocity and a Poisson-type equation for the pressure at each time step. The fast tensor-product solver presented here could function as a perfect preconditioner for these elliptic solves; for small free surface deformations, the convergence should be very rapid. The solver should also be insensitive to the potentially large aspect ratios associated with the computational domains for this class of problems. Note that the two-dimensional cross-sections may be quite general and may not be possible to express in tensor-product form [11].

Future work will also extend the approach presented here to include fast tensor-product solvers for a combined space-time treatment. In a second paper (Part II), we will discuss a tensor-product solver for the pure spectral case. This is part of an ongoing research effort to extend the possibilities for parallel processing in the time direction, thus allowing for an overall increased speedup for the simulation of evolution problems described by partial differential equations.

Finally, we mention an open area for the possible application of fast tensor-product solvers, namely, the approximate solution of the Boltzmann equation. For this type of problems, we have 6 independent variables in three physical space dimensions: 3 velocity directions and 3 physical coordinate directions. A "cross-section" in the xyz-"plane" is here invariant with respect to all the velocity directions. This could potentially be exploited in the construction of tensor-product bases and fast solvers.

References

[1] Bjørstad, P.E. and Tjøstheim, B.P. (1997). Efficient algorithms for solving a fourth-order equation with the spectral Galerkin method. *SIAM J. Sci. Comput.*, **18**(2), 621-632.

[2] Carvalho, M.S. and Scriven, L.E. (1999). Three-dimensional stability analysis of free surface flows: Application to forward deformable roll coating. *J. Comput. Phys.*, **151**(2), 534-562.

[3] Chu, D., Henderson, R., and Karniadakis, G.E. (1992). Parallel spectral-element-Fourier simulation of turbulent flow over riblet-mounted surfaces. *Theoretical and Computational Fluid Dynamics*, **3**, 219-229.

[4] Couzy, W. and Deville, M.O. (1995). A fast Schur complement method for the spectral element discretization of the incompressible Navier-Stokes equations. *J. Comput. Phys.*, **116**, 135-142.

[5] Fischer, P.F. (1997). An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. *J. of Comput. Phys.*, **133**, 84-101.

[6] Golub, G. and Van Loan, C.F. (1983). *Matrix Computations*, John Hopkins.

[7] Henderson, R. (1997). Nonlinear dynamics and pattern formation in turbulent wake transition. *J. Fluid Mech.*, **353**, 65-112.

[8] Koschmieder, E.L. (1993). *Bénard Cells and Taylor Vortices*, Cambridge University Press.

[9] Kwan, Y.-Y. and Shen, J. (to appear). An efficient direct parallel spectral-element solver for separable elliptic problems. *J. Comput. Phys.*, DOI: 10.1016/j.jcp2007.02.013.

[10] Lynch, R.E., Rice, J.R., and Thomas, D.H. (1964). Direct solution of partial differential equations by tensor product methods. *Numer. Math.*, **6**, 185-199.

[11] Medale, M. and Cerisier, P. (2002). Numerical simulation of Bénard-Marangoni convection in small aspect ratio containers. *Numer. Heat Transfer, Part A*, **42**, 55-72.

[12] Maday, Y. and Patera, A.T. (1989). Spectral element methods for the Navier-Stokes equations. In Noor, A.K. (ed.), *State of the Art Surveys in Computational Mechanics*, ASME, New York, 71-143.

[13] Patera, A.T. (1986). Fast direct Poisson solvers for high-order finite element discretizations in rectangularly decomposable domains. *J. Comput. Phys.*, **65**, 474-480.

[14] Shen, J. (1994). Efficient spectral-Galerkin method I. Direct solvers for the second and fourth order equations using Legendre polynomials. *SIAM J. Sci. Comput.*, **15**(6), 1489-1505.

[15] Toselli, A. and Widlund, O.B. (2004). *Domain Decomposition Methods - Algorithms and Theory*, Springer Series in Computational Mathematics, **34**.

[16] Tufo, H.M. and Fischer, P.F. (1999). Terascale Spectral Element Algorithms and Implementations. Gordon Bell Prize paper, Proc. of the ACM/IEEE SC99 Conf. on High Performance Networking and Computing. IEEE Computer Soc., CDROM.